A transversal alignment between measurements and enterprise architecture for early verification of telecom service design

Iyas Alloush, Yvon Kermarrec, Siegfried Rouvrais

A Transversal Alignment between Measurements and Enterprise Architecture for Early Verification of Telecom Service Design

Iyas Alloush[1,2], Yvon Kermarrec[1,2], and Siegfried Rouvrais[1,3]

1: Telecom Bretagne, Institut Mines-Telecom, Université européenne de Bretagne
Technopole Brest Iroise, CS 83818 29238, Brest Cedex 3, France
2: UMR CNRS 6285 Lab-STICC
3: IRISA
firstname.lastname@telecom-bretagne.eu

Abstract. Early verification of Telecom Services (TS) at the design time helps an enterprise to avoid wasting of implementation cost and time. Simulation provides the designer with helpful feedbacks on TS design leaving the implementation and installation costs behind. Our objective in this paper is to present our approach to obtain valuable feedbacks from network simulators and relate them to the information from high abstract scenario of a TS. Relying on Model Driven Engineering discipline and Enterprise Architecture standard, we propose to associate measurement elements with the design ones of different abstraction levels. We implement a model transformation to generate automatically the configurations for NS-3 simulator as a test-bed for our case studies. We illustrate our approach with a video conference example, presenting the different abstract levels and their relationship with measurements.

Keywords: Telecommunication Service, Enterprise Architecture, Model Driven Engineering, Tool Chains, Simulators, Measurements, NS-3, OPNET, Early Verification, Model Transformation, Code Generation

1 Introduction

The production of Telecommunication Services (TS) is a long term process and consumes many resources. A mistake in the TS design may result in serious consequences to the rest of the life cycle activities. It is difficult and cost expensive to fix errors and flaws in the design after the implementation activity, especially if there are hardware elements to be installed. Therefore, we will take into consideration two points: the TS characteristics, and the TS creation activity.

The TS have several characteristics that make them different from other software systems. The platforms that a TS relies on are large scaled [19] and used by enormous number of customers. The TSs themselves are pervasive [4] and context aware [5] in many cases. A TS may rely on software and hardware elements to perform its tasks. Normally, the TS is composed of numerous number

of applications that interact with each other, this results in a complex TS design [9]. Describing TS designs using Enterprise Architecture (EA) helps in managing the complexity problem [14].

The contribution of this paper is in the scope of TS creation activity (TSC) [13] and relies on a recent dissertation [11]. We address the early verification activity right after the design one, and before the implementation phase. The aim of the verification activity is to "check that the software meets its stated functional and non-functional requirements" [21]. The Service Creation Environment (SCE) [6] is a collection of software tools that rely on reuse infrastructure used according to the service development methodology, to assist the service developer(s) by automating and simplifying the Service Creation process.

In our research we face the challenges: (1) improving Quality of Service (QoS) that meets the customer satisfaction, and some performance non-functional requirements (NFRs) that concern the service and network providers such as CPU and memory loads; (2) the time to market that should be as short as possible; (3) improving the cost efficiency.

Our aim is to assist the different stakeholders that are involved in the TS specification activity, taking into consideration the three factors mentioned recently. In this paper, we will answer to the following research question: **How to enable the estimation of the Quality of Service (QoS) violations and other possible performance non-functional requirements in the TS design using network simulators, and relying on the different abstraction levels of a TS architecture?**

Our objective in this paper is to propose a centric solution that enables the TS designer and developer to obtain valuable feedbacks on the design. This will help the TS designer/developer to improve the quality and to detect the errors or flaws of the design.

Our approach is to bring the measurement element from the operation activity and insert it into the design activity, so to enable the observation of the TS virtual execution through network simulators. Our first contribution is to propose a transversal alignment between the measurement and design elements by extending the design meta-model that was proposed in [11] to include measurement and probe elements. The second contribution is to map between the design and both of simulation Technical SPace (TSP) and Java Virtual Machine (JVM) taking a case study of NS-3 as a network simulator, where we intend to aggregate the measurements from the different architectural elements related to the upper layers of ArchiMate [22].

We illustrate our approach with an example of a TS: video conference, presenting the generated code according to the design and measurement configurations. This example relies on a previous work done in [11,7].

In section 2, we highlight the Enterprise Architecture (EA) with its architecture, and show the benefits from applying its structure to our approach. In section 3, we will provide an explanation about Model Driven Engineering (MDE) and its role in our approach, and we explain model transformations as we rely on code generations to generate the code. Section 4 presents in details our ap-

proach to answer to the mentioned research question. In Section 5, we present a short explanation about network simulators, and show how do we map between design and simulation technical spaces. Thus, we show our work to create a reliable test bed for our current and future experiments. In Section 6, we present an example of obtaining data that are related to different layers of EA using to different technical spaces (simulation and JVM). In section 7,we present some of the related work, highlighting the points of interest to our objective. Then we conclude in section 8 and discuss our future work.

2 Enterprise Architecture and ArchiMate

An **Enterprise Architecture (EA)** [14] "is an instrument to articulate an enterprise's future direction, while also serving as a coordination and steering mechanism toward the actual transformation of the enterprise".

EA framework facilitates dealing with complex architectures, as it applies the different viewpoints of the different stakeholders [20] (e.g. service designer, service developer, customer, etc) in order to capture all the aspects of the enterprise objectives.

Thus, we are interested in the underlying platform and business viewpoints. When the TS design is to be implemented, there are two types of models that compose the TS: Platform Independent Model (PIM) and the Platform Specific Model (PSM). PIM contains the components that are independent from the execution platform, while the PSM contains elements of the execution platform and their functions.

We apply ArchiMate [22] which is an EA modeling language. It decomposes the system as the EA does according to different viewpoints. Additionally, its profile contains means that help the TS designer, as they are suitable for the IT domain. They define behavioral and structural elements connected to each other by wide range of relationships. This layered architecture allows changes on the design to be independent from other layers [21], and makes finding an error or design flaw much more easier.

In [11,7], ArchiMate profile is extended to define the Domain Specific Modeling Language (DSML), where different elements that are related to TS domain are included in the extended profile. For the verification activity (Fig. 3), we extend ArchiMate abstract syntax to include measurements, simulation and analysis tools with their relationships that are inherited from ArchiMate definition [22].

A TS designer should verify the TS through its architecture that is composed of layers of different levels of abstraction (PIMs and PSMs). In this paper, we present our approach to connect measurement elements to these different layers.

3 Model Driven Engineering and Model Transformations

System modeling [17]: "refers to an act of representing an actual system in a simple way". With modeling, the parameters of the system can be modified, experimented, and analyzed. In our approach (Fig.3), models plays a central role

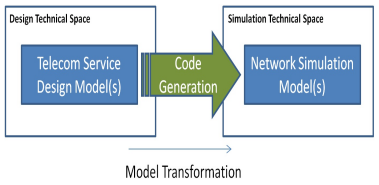

Fig. 1. Applying MDE to bridge between Design and Simulation modeling

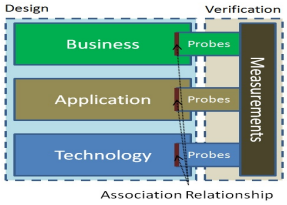

Fig. 2. A transversal Alignment Approach between Measurements and Design

to represent systems of different technical spaces: Design, and Simulation. MDE is well supported by Eclipse IDE [1]. Eclipse contains powerful tools for MDE such Eclipse Modeling Framework (EMF) which forms the core of our modeling environment in the design Technical SPace (TSP).

Model transformation (MT) is one of the key concepts in MDE, and in our approach. MTs offer automated way to create new models from given ones according to the mapping rules that controls them.

XPAND [2] is a model-to-text transformation language that makes it possible to iterate over input model instances to generate text files. This enables us to generate executable and simulation codes for Platform Independent Models (PIMs) [11] and Platform Specific Models (PSMs) [7].

4 Measurement and Design Transversal Alignment Approach

The TS designer needs to get feedbacks for his design related to errors/flaws or quality violations. We have recently associated the measurement model to the elements of the technology layer that represents the underlying platform in our approach, and its model is a PSM. We have selected the IMS core network platform to be represented in this layer [7,12]. Such association will provide a TS designer feedbacks that are related to the network level, which is very low in abstraction. This leads us to a question: *How can we perform measurements on elements of higher level of abstraction?*

We use ArchiMate language (section 2). It contains the business layer that provides the ability to relate the business actors from an enterprise (e.g. user) with the different types of behavioral elements (such as business interactions, events, functions, etc) in the system and in high level of abstraction [22]. Thus, ArchiMate business layer can cover both of the (Use Case and Scenario [21]) representation fundamentals. ArchiMate language provides relationship between every two neighboring layers [22] in abstraction (e.g. Business and Application layers), which makes it possible to exchange events and interactions between the 3 layers of ArchiMate.

We propose to enable the association between the measurement element (the Probe) and all behavioral and structural elements (Fig. 2) of ArchiMate on the Meta-Model. The measurements are aligned transversally with ArchiMate layers and their probes are following the layering concept of ArchiMate (Fig. 2), thanks to the one-to-many aggregation relationship between the Measurement

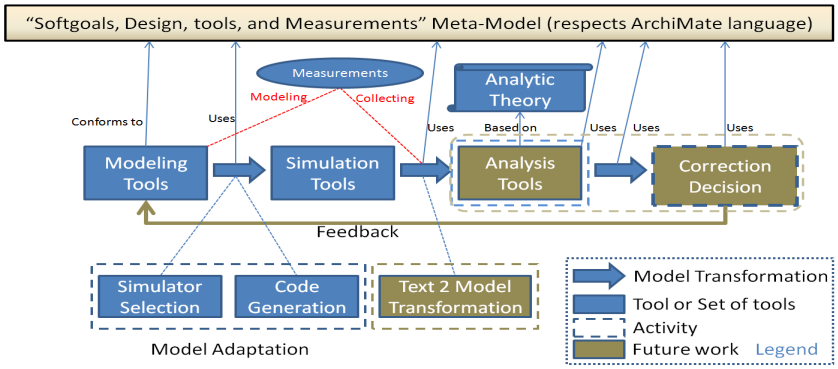

Fig. 3. Verification of Telecom Services relying on tool chains

and Probe(s) in the Meta-Model (Fig. 4). Respecting the ISO-Square standard [8], we propose the measurement meta-model as composed in the following structure (Fig. 4): (1) Measure: Represents the output value of the measurement; (2) Probe: Represents an entity that captures the data when an event is triggered (e.g. a packet is dropped), an action is performed, or during all of the simulation time. The probe concept is inspired from OPNET [1] structure [10]. The probe element is associated with structural and behavioral entities from ArchiMate profile. This provides the ability of collecting measurements from elements of the different layers of EA. Probes have different types due to the nature of the data type that they can obtain (e.g. numerical, time). In the verification activity, one may use all the probes that are related to one measurement in one layer of EA, or distribute them between different layers. In this paper, we will consider the probes per layer case, leaving the other one to future work; (3) Probe Function: The probe function defines the procedure (how) to capture the data; (4) Measurement Function: Represents the function (how) to aggregate the data collected by the different probes.

The measurement meta-model (MM) that we propose (Fig. 4) makes it possible to use the measurement per layer or as a multi-layer measurement, where the probes may be associated with the elements of one layer or distributed between the different layers. In this MM, the layer and data type attributes are clear in the probe element. Thus, the probes are aligned with the design elements in the same layers, while the measurement elements are aligned with the EA architecture transversally (Fig. 2) as they collect the data from the different probes thanks to the aggregation relationship between them (Fig. 4).

This meta-model (Fig. 4): (1) Makes it possible to tune the granularity level of the data collection during the verification of the TS design; (2) Classifies the measurements according to the viewpoint of the stakeholder and to the different domains that are used in an enterprise. So we can have business, application, and technology classes of measurements. This can help in managing the classifica-

[1] OPNET is a wide-used network simulator, it can handle different network elements of both hardware and software nature. `www.opnet.com`

tion of QoS requirements [18]; (3) Enables earlier selection of the measurements knowing the requirements from the business layer. Knowing the set of measurements needed for the technology layer, enables us to select the proper network simulator.

5 Network Simulators

We need to find a proper approach to execute virtually and analyze the design models. There are different approaches for modeling a system for experimentation purposes: (1) Simulation Approach: In the network simulation domain, Discrete-Event Simulation (DES) is a powerful research technique to investigate protocol designs, interactions, and large-scale performance issues [16]. Simulation allows researchers to experiment their prototypes with low cost and implementation time than dealing directly with real network elements. Simulators provide a TS designer with valuable observations and measures that helps in detecting design errors/flaws and quality violations; (2) Analytic Approach: Analytical modeling approach is to model the system mathematically using applied mathematical theories (e.g. Queuing and Probability theories).

In our approach, we rely on simulation approach. Modeling using EMF in eclipse (section 3) differs from modeling the same design using the simulation program language. We can map each element from the design TSP to the corresponding one in the simulation TSP using model transformations (section 3).

The objective from any network DES program is to run a scenario that contains structural (e.g. nodes) and behavioral (e.g. send message function) elements for a previously-set simulation period, in order to obtain traces and logs at the end of the simulation run. These logs can be then analyzed later to find the possible errors/flaws or quality violations in the design.

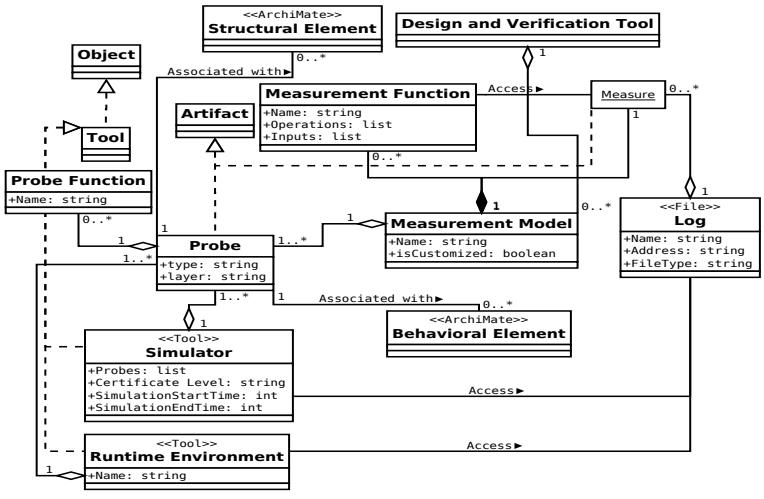

Fig. 4. Measurement view of the meta-model

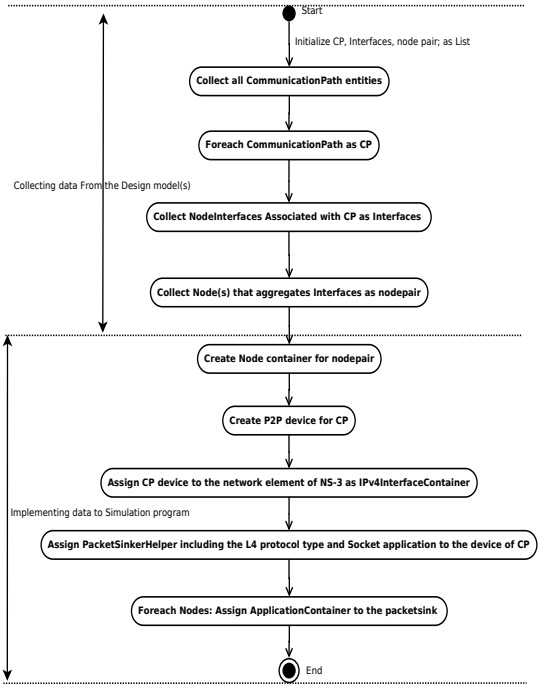

Fig. 5. The activities to map design structural entities to NS-3 simulation program

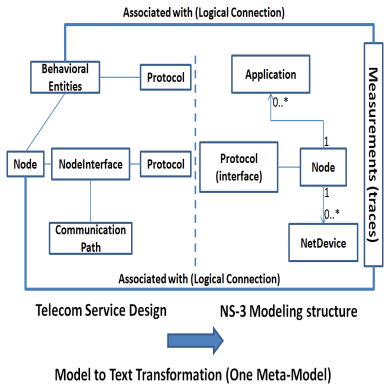

Fig. 6. Mapping between Design model and NS-3 simulation program

Our approach takes into consideration the structural and behavioral elements of the TS design. The challenge here is to find the simulator elements that correspond correctly to the design elements, and to implement the simulation program in the right way. Every element from the design model should be mapped to the simulation program. Such a mapping needs experience in both areas: TS design, and simulator specifications, this can be considered as a limitation in our approach. Because implementing the mapping rules in the model transformation template (e.g. XPAND template, section 3) needs the mentioned expertise and a considerable time to be sure that no design elements were missed, also to insure that the language rules of the simulation program were respected.

5.1 NS-3 Simulator: A test bed for experiments

Network simulators help to estimate the TS performance and QoS metrics. But they differ from each other in their capabilities, certificate level, graphical interfaces, granularity levels, and built-in integrated tools (e.g. animators, report generators, analyzers, etc).

Regarding to the capabilities, we are interested in the measurement capability to differentiate between the different network simulators. There is an important point in NS-3 [3] is that the whole simulation program can be implemented in one C++ file in NS-3, where all of the simulation parameters can be configured including generation of traces and logs [3]. This feature in NS-3 enables us to

generate code automatically (according to section 3) for measurements aggregation in addition to the scenario implementation and its underlying platform (e.g. IP Multimedia Subsystem (IMS)). C++ has different functions that assist plain-text output, thus we can include information from the design combined with the measurement results in the output files.

Additionally, NS-3 supports different types of easy-configured applications [3] (e.g. Socket using TCP or UDP). Unfortunately, NS-3 does not support IMS core network model yet. Taking advantage from the advanced models of the application layer provided in NS-3 [16,3], we call the functions of IMS in a high abstract level. There is a function called (SendTo) in our technology layer MM [7] that represents the message sending procedure. Thus, we generate the SIP messages as UDP or TCP packets using socket application that is already deployed in the application layer of NS-3 model. More details about the models of NS-3 one may refer to [16].

In relation to the structural entities of IMS platform that is presented in [7], a mapping between these entities (e.g. Nodes, CommunicationPaths, NodeInterfaces, etc) was performed to create the underlying platform where the scenario is going to run. Our mapping activities start with collecting data from the design model and then map them to the target simulator. The collection procedures start from the CommunicationPath as there is only one CommunicationPath links between every two linked nodes, this constraint is defined in the meta-model of the technology layer of [7]. Then we iterate to query for the interfaces that are involved in this communication until we reach the node pairs... (Fig. 5). The steps to implement code in NS-3 appears in the same figure. Figure 6 presents the architecture of both technical spaces: Design and NS-3 simulator. The activities in (Fig. 5) show that we rely on collections and iterations without any exceptions and respecting the types and constraints of the meta-model. This should confirm that every design element is covered when mapping to the simulation TSP.

6 Example on Measurements associated with Video Conference TS: NS-3 and JVM case study

There is a major difference between the technical spaces of network simulation such as NS-3/OPNET and general purpose languages (GPLs) such as Java. Our example in this paper illustrates the measurement correspondence to the different layers of the design architecture described in ArchiMate language. We will give an example of two layers: the business and technology. We generate Java code from the business layer, and simulation code from the technology one following the description in section 5. The Java code generation relies on the model transformation rules that were defined in our previous work in [11].

We rely on a scenario (Fig. 7) for video conference which is modeled using a Domain Specific Modeling Language (DSML) [11]. This scenario is represented in the business layer of ArchiMate, and shows the correspondence of business actors with the different activities that compose it.

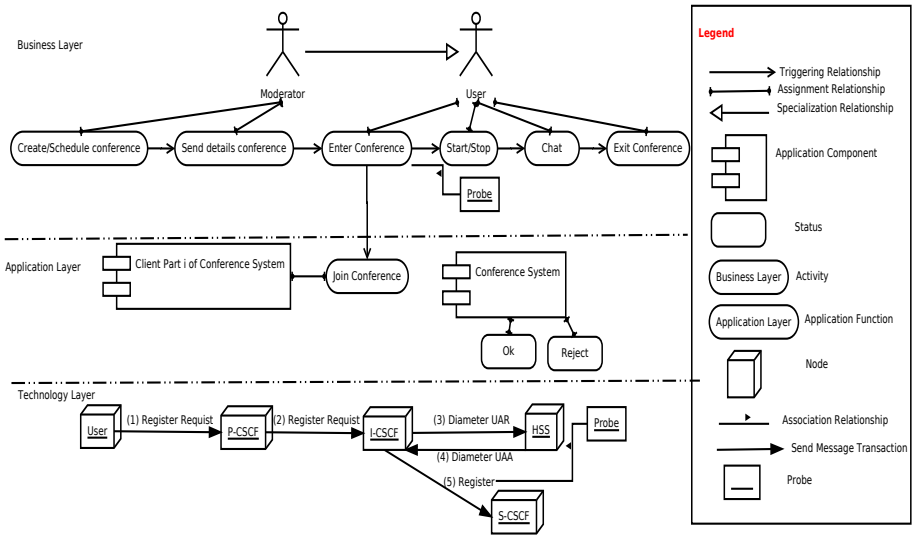

Fig. 7. Video Conference models inspired (with modifications) from [11]

6.1 Associating probes with the elements of the Business Layer

Associating probes with the business layer elements gives the ability of obtaining data from the business domain such as user-related information. Figure (7) shows that when the user enters to the conference system joining the group, a function in the application layer called join conference is triggered (called) by the (Enter Conference) activity. This application function triggers a registration procedure in the technology layer, to register the user into the conference service in the simulation technical space.

Figure (9) presents the Java code that is generated by the MT rules for the business layer elements. It shows how the probe function (getName) is called inside the method (enterconference), where this function is not modeled to be triggered by the activity (Enter Conference) in (Fig. 7). This illustrates an example for our method in obtaining the data from different elements of Business Layer relying on our approach. One can also find in (Fig. 9) that there is a method named (setName) which is used to set the name of the customer during the runtime execution, and it should be called by the running system such as a mobile device.

6.2 Associating probes to the elements of the Technology Layer

Associating the probes with elements from technology layer gives the ability to obtain different type of measurement data from the elements of the underlying platform (IMS in our research [7]). For the example (Fig. 7), we generate the simulation code that is used to configure NS-3 as a case study. We rely on the video conference example presented in [7] to generate the configuration script. The association relationship between the probe (named (RegistrationTime) of

type (Time) and layer (technology)) is used to capture the registration time of the user when he joins the conference system. We set the ProbeFunction method value to (Simulator::Now()) (Fig. 8), which is a command used to obtain the current time during the NS-3 simulation runtime.

There is still a question that needs to be answered from the previous example about the aggregation between the results of the probes, which are attached to different TSPs (e.g. Simulation and JVM). We consider this problematic as a future work.

7 Related work

We consider the following aspects during the review: (1) The ability of collecting measurements from structural elements as well as the behavioral ones; (2) Linking automatically the design modeling and Simulation or other execution software technical spaces; (3) Domain Specificity of the approach; (4) The ability to obtain data after executing the design and on high level of abstraction including the scenarios.

- In [15], the paper describes a model-based approach to create telecom services. They implement the service relying on IP Multimedia Subsystem (IMS) infrastructure and using code generation techniques from the MDE discipline. Using Eclipse software, they could close the abstraction gap between the design and the execution environment, by hiding the low level protocol and architecture details. Although their approach considers the domain specificity and high abstraction level of the design models, but no measurement capture method appears in their approach;
- The paper [23] presents work that is very close to the formalism which we use. The authors apply MDE and rely on DEVS (Discrete EVent Systems Specification) formalism to smooth the modeling and simulation cycle and facilitate the coupling of models using heterogeneous formalisms. They provide the SimStudio software which aims at providing modeling and simulation tool chain. Their approach and ours depend on meta-models that contains different views to the different activities of the tool chain (modeling, simulation). Their approach doesn't have support for structural elements representation while limited to the behavioral ones. Their approach seems to depend on one abstraction level of measuring capability, while we use probes to obtain data from different layers of EA;
- The work presented in [6] addresses the same scope of service creation activity. They propose a methodology that takes into consideration the customer and service provider requirements. They apply code generation to produce Java and C++ codes directly from these PIMs, with no mention to simulation tools;
- In [5], the authors work on context aware services creation process. Not far from our approach, they link between the knowledge modeling and implementation activities, relying on model transformations. Their approach doesn't seem to rely on simulators but on runtime environments such as JVM. Their approach takes the end-user only into consideration, where we

```
void f_RegistrationTime_64274c95 (Time oldval, Time newval){
        std::cout << "Traced" << oldval
         << "to" << newval << std::endl;
        std::cout << "Traced:"
        << Simulator::Now()<<std::endl;
        NS_LOG_INFO ("The probe RegistrationTime_64274c95 is set from the value "
        << oldval << " to the value " << newval);
}
void f_DevQueueDrop_64274c93 (uint64_t oldval, uint64_t newval){
        std::cout << "Traced" << oldval
        << "to" << newval << std::endl;
        NS_LOG_INFO ("The probe DevQueueDrop_64274c93 is set from the value "
        << oldval << " to the value " << newval);
}
```

Fig. 8. Probe function implemented in NS-3 to capture the registration time of a user

```
public String getName() {
    return name;
}
public void setName(String newName) {
    this.name = newName;
}
public void exitconference() {
}
public void enterconference() {
    Clientpartiofconferencesystem clientpartiofconferencesystem
            = new Clientpartiofconferencesystem();
    clientpartiofconferencesystem.joinConference();
    this.startStop();
    this.getName();
}
```

Fig. 9. User Class Java code-Business Layer

take service designer and user models into account thanks to the EA standard. Their approach respects domain specificity and data capturing during and after the execution, but it is only from high abstract models.

8 Conclusion and Future work

In this paper, we have presented our work in the activity of TS verification, by defining a new meta-model that links measurements to the design. We have implemented a test-bed based on NS-3 network simulator to rely on for case studies. Our approach is to link the measurements to the different elements of the TS design as aligned transversally with the EA framework. We have illustrated our approach with example on the video conference service, presenting the code that is generated from models of different abstraction levels. This provides a centric solution that enables the TS designer to obtain valuable feedbacks on the design so to improve the qualities and detect possible design errors or flaws.

Our approach, as presented, assists the different stakeholders during the design activity thanks to EA and ArchiMate. It is capable to implement structural and behavioral entities of the design in the simulation technical space, while implementing the required measurements automatically as well. Raising the measurements' association to the layers higher than technology (PSM) one in the abstraction level makes it possible to obtain data from the PIMs directly. This enables us to link goals, design, and verification elements together.

On the other side, the number of inputs of the measurement function is limited to two in our implementation. The implementation of code generation templates is a time consuming activity and needs experience in both simulation and modeling domains.

For the future, we intend to answer to the problematic that we raise in section 6, where aggregating the data obtained from different design layers is important but still challenging. Additionally, we intend to analyze automatically the measurements resulted from the simulation activity to check the goals' satisfaction.

References

1. Eclipse IDE, `http://www.eclipse.org/`

2. Eclipse modeling, http://www.eclipse.org/modeling/
3. NS-3 Manual-Release ns-3 (November 30 2012)
4. Achilleos, A., Yang, K., Georgalas, N., Azmoodech, M.: Pervasive service creation using a model driven petri net based approach. In: Wireless Communications and Mobile Computing Conference. IWCMC '08. International. pp. 309 –314 (2008)
5. Achilleos, A., Yang, K., Georgalas, N.: Context modelling and a context-aware framework for pervasive service creation: A model-driven approach. Pervasive and Mobile Computing 6(2), 281 – 296 (2010)
6. Adamopoulos, D., Pavlou, G., Papandreou, C.: Advanced service creation using distributed object technology. Communications Magazine, IEEE 40(3), 146 –154 (march 2002)
7. Alloush, I., Chiprianov, V., Kermarrec, Y., Rouvrais, S.: Linking telecom service high-level abstract models to simulators based on model transformations: The ims case study. In: Information and Communication Technologies, vol. 7479, pp. 100–111. Springer Verlag (2012)
8. Azuma, P.M.: Iso/iec cd 25000.2 - software and systems engineering – software product quality requirements and evaluation (square) (2003)
9. Bowen, T., Dworack, F., Chow, C., Griffeth, N., Herman, G., Lin, Y.: The feature interaction problem in telecommunications systems. In: SETSS (1989)
10. Chang, X.: Network simulations with opnet. In: Simulation Conference Proceedings, 1999 Winter. vol. 1, pp. 307 –314 vol.1 (1999)
11. Chiprianov, V.: Collaborative Construction of Telecommunications Services. An Enterprise Architecture and Model Driven Engineering Method. Ph.D. thesis, Telecom Bretagne, France (2012)
12. Chiprianov, V., Kermarrec, Y., Rouvrais, S.: Extending enterprise architecture modeling languages: Application to telecommunications service creation. In: The 27th Symposium On Applied Computing. pp. 21–24. ACM, Trento (2012)
13. Combes, P., Renard, B.: Service validation. Computer Networks 31(17), 1817 – 1834 (1999)
14. Greefhorst, D., Proper, E.: Architecture Principles, The Enterprise Engineering series, vol. 4. Springer (2011)
15. Hartman, A., Keren, M., Kremer-Davidson, S., Pikus, D.: Model-based design and generation of telecom services (2007)
16. Henderson, T.R., Roy, S., Floyd, S., Riley, G.F.: ns-3 project goals. In: Proceeding from the 2006 workshop on ns-2: the IP network simulator. WNS2 '06, ACM, New York, NY, USA (2006)
17. Issariyakul, T., Hossain, E.: Introduction to Network Simulator NS2 (2009)
18. Lee, C., Lehoczky, J., Siewiorek, D., Rajkumar, R., Hansen, J.: A scalable solution to the multi-resource qos problem. In: Real-Time Systems Symposium, 1999. Proceedings. The 20th IEEE. pp. 315 –326 (1999)
19. Nicol, D.M., Liljenstam, M., Liu, J.: Advanced concepts in large-scale network simulation. In: Proceedings of the 37th conference on Winter simulation. pp. 153–166. WSC '05, Winter Simulation Conference (2005)
20. Simonin, J., Le Traon, Y., Jezequel, J.M.: An enterprise architecture alignment measure for telecom service development. 11th Ieee International Enterprise Distributed Object Computing Conference, Proceedings pp. 476–483 (2007)
21. Sommerville, I.: Software Engineering. PEARSON, 9 edn. (2011)
22. The Open Group: ArchiMate 1.0 Specification (2009)
23. Touraille, L., Traoré, M.K., Hill, D.R.C.: A model-driven software environment for modeling, simulation and analysis of complex systems. In: Proceedings of the Symposium on Theory of Modeling & Simulation. pp. 229–237 (2011)